22 | 学习观：程序员如何定义自己的技术舒适区？

2020-07-06 臧萌

职场求生攻略 进入课程 >

讲述：臧萌
时长 13:01 大小 11.93M

你好，我是臧萌。从这一节开始，我们就进入到了技术成长篇。不过在这篇里，我们不去深入聊任何具体的技术细节，而是围绕技术这个核心，和你聊聊如何看待技术，聊聊围绕技术的那些事儿。这是技术成长篇的第一篇文章，我来带你一起看看，我们应该如何定义自己的技术舒适区。

程序员的舒适区

平时我们谈论所处的工作状态的时候，都喜欢用舒适区、学习区、恐慌区这三个词来形容。我觉得这几个词形容得非常贴切。一个人处在舒适区，就是可以轻松愉快地应对日工作，闭着眼睛都能把工作完成。学习区呢，就是工作有一点挑战，需要学点东西，但是也不需要费什么劲儿。恐慌区，就是看啥啥不懂，干啥啥不会，满满的压力，完全无法进行工作。

不要相信舒适区

其实很多职业都可以轻松打造出自己的舒适区。比如出租车司机、售货员、手机贴膜员这些职业，从技术技能角度来讲，花个几个月，就可以打造出自己的舒适区，然后就可以轻轻松松完成自己的工作了。

但是程序员这个职业不同。程序员所在的软件行业是一个高速发展的行业，如果要把技术当做安身立命的本事，那么程序员就必须要一直学习知识，才能保证自己不被淘汰。下面我来说说为什么。

假设现在你可以不太费劲地完成自己的工作，紧接着的一段时间，你也不需要学什么东西，也可以完成自己的工作，看似岁月静好，十分惬意。但是设想一下，在 IT 这么一个飞速发展的行业中，为什么没有给你所做的工作带来新的挑战？为什么不需要你们的技术进行升级换代？

答案很可能就是你所在的公司，或者现在所从事的职业，已经处于淘汰、停滞不前的状态了。公司业务非但没有增长，可能还缩水了。这时候一切舒适、惬意、轻松所营造的岁月静好的氛围，其实都是表象。而表象下面，则是你很可能被淘汰。

技术要么发展，要么被淘汰

急速发展的软件行业，尤其是互联网行业，是一个不进则退的行业。业务量和业务的复杂度都在增长，留给开发的时间则是越来越短。为了满足这一切，技术的更新换代，甚至技术方向的完全改变，都是不可避免的。

比如微服务架构，一个重要的目标是能解耦依赖，进而加快迭代。本着配置简单、从轻从快的思想的 SpringBoot，很好地满足了微服务的要求，让原本部署在 J2EE 容器里的 war 包，变成了返璞归真的 jar 包，可以直接运行，吃过的都说香。

最近几年风行的 Docker，就颠覆了之前打包部署的方式，让打包部署上线这个过程简单又标准。同时与之一起的虚拟化技术，也颠覆了计算资源分配和管理。当然还有很多很多别的技术，比如熔断降级，监控等等，这些技术串在一起，犹如一条龙服务一样，解放了我们程序员的生产力，让我们可以更快地对业务需求作出响应。

如果一个程序员现在还在使用 Tomcat、web.xml、打包、物理机、拷贝部署 war 包，一方面说明公司对程序员的效率要求不高，业务压力不大，也就是业务发展的不好。另一方面，如果这个程序员想出来找工作，和面试官一聊，好么，考古级的技术栈，请问谁愿意要这样的程序员？

我们不能拿公司用不到做借口，程序员就是要有学习的本能冲动，推动公司的技术进步，至少要让公司的技术能跟得上行业标准水平。所以，就好像北上广不相信眼泪一样，我们程序员也不要相信有舒适圈的存在。如果一个工作让一个程序员享受到了舒适圈待遇，那么就说明这个工作可能要凉，可以考虑换个工作了。

学到手的技术会贬值

其实从利益的角度来讲，我们程序员也不应该呆在所谓的舒适区。一个客观事实是，软件行业的技术在飞速地发展，也就是说，程序员现有的技术知识存量，会一直贬值缩水。

长期来看，你觉得公司会为贬值的技术付出不变的，甚至还要增加的薪水吗？如果人才市场上，有性价比更高的人才，要的薪资更少，能做的事情更多，你觉得公司会不会考虑换人呢？

所以说，也许老的技术应对公司的业务足够了，但是这并不代表一个程序员就可以守着老的技术，在一个公司一直混下去。只要技术还在进步，那么就会有人可以用更少的工作经验，更低的薪水，更好地完成一份工作。别等到自己的看家本领被淘汰了，才想起来学习。

做一个合格程序员，就要拒绝技术舒适圈。但是拒绝舒适圈，不代表不能有一个舒适的状态。下面我们就来聊聊程序员的舒适状态。

程序员的舒适状态

程序员的生涯中，只有一直学习才是最舒适的状态。这里说的舒适当然不是指轻松随意，而是通过不断的学习，让自己可以轻松地应对工作，知道自己能够一直学得会、跟得上最新的技术。而自己的这种状态，可以带来发自内心的自信。这种技术自信，就是让自己舒适的根本。那要如何建立这种技术自信呢？我们接着往下讲。

找到提供正反馈的学习目标

其实关于学什么这个问题，我也走过弯路。立过很多 flag，最后大都啪啪打脸。捂着“打肿”的脸，我就开始反思，为什么没坚持学下去呢？最后我得出了一个结论：学习没有形成正反馈。

没有正反馈，就是指学了之后找不到用的地方，没有看到学习带来的好处，没有感觉到自己在成长，热情就开始大大下降。最后，心里随便找个什么借口，比如工作很忙啦、这个东西用不到啦之类的，也就弃坑了。

最后，我发现我是一个比较懒的人，也发现工作确实让人“急功近利”，很难像学生时代那样心平气和地为学而学。后来，我总结出来一个寻找学习目标的法则：没事儿不找事儿，遇事儿不怕事儿。

没事儿不找事儿，就是当遇到对你没太大帮助的技术，不去投入大把精力学它。只去简单了解这个技术的大概，细枝末节不去深究，只要知道它解决的问题是什么就可以了。

这样做有两个原因，一方面，当一个技术名词频繁出现在你眼前的时候，不能完全无视它。我们得先知道它是干嘛的，通过什么方法和技巧，解决了什么问题。不到这一步，你就是固步自封了。接下来，你就得想想在自己的兴趣和工作的范围之内，这个技术能否用得上，用上了是否能带来好处。如果不能给你带来好处，那就先不去管它。

遇事儿不怕事儿，就是说每当你遇到任何需要解决的技术问题，不要绕着走，要记下来，挤时间吃透它。因为根据我的经验，你遇到过一次的问题，就会遇到第二次第三次。与其让问题一遍遍地欺负自己，不如自己主动点，第一次遇到问题的时候就别认怂，直接学起来。这样的好处就是正反馈来得很直接，成果立竿见影。

对比上一条说的“没事儿不找事儿”，如果你工作中遇到问题或者某个细节不清楚的情况，正愁不知道学啥呢，忽然一个问题“千里送人头”。这时候还等啥，送上门的不收割，难道还放走了不成？

在这里我举个自己的例子。大概在 10 年左右吧，Hadoop 在国内已经开始火了。我的心也躁动得不行，大数据、分布式、Google 论文这些词儿都在撩着我。于是我就立了一个 flag：“今年学 Hadoop”。结果毛都没学，只是知道了一个 HDFS 和 Mapreduce 的基本概念，一丁点实操都没有。

忙吗？那一年其实不算忙。就是自己懒，而且内心就知道这玩意学了用不上。后来过了几年，我换了工作，有一个工作要用到 Hadoop 和 HBase，从把集群折腾起来了，到用客户端 API 跑通 POC，总共也没用俩星期，期间还做了别的很多工作。

无数的经验告诉我，有需求的情况下，知识的供给才是有效的供给。知识解决了问题，就有了正反馈，学习的飞轮才能持续地转下去。还有就是人的潜力是很大的，deadline 才是第一生产力，哈哈。

有效的学习姿势

前面说的方式也有一个问题，就是可能无法系统地学习知识。如果只是东一榔头西一棒槌的学，知识会很不成系统。这样不是一个好的学习姿势。我总结出一个学习的方法，就是主动花功夫构建骨架，被动增加知识点。

主动花功夫构建骨架，就是指有大的方向不会，就选个趁手的资料，有计划成系统的学习。任何一门新技术，都要先经历这个过程。比如 Spring，核心就是管理对象的创建和依赖关系。你得深入了解 Spring 内部机制，了解 Spring 是如何做到这一点的。

如果再往细了说，可以从以下五个点来构建骨架。

核心架构设计：这个技术有哪些核心的架构设计。架构设计决定了技术的特征，也决定了这个技术的优劣势和适用的场景。比如 Java 是基于字节码和虚拟机的，HBase 的本质就是一张排序的大表，Aerospike 本质就是一张大的散列表等等。

功能模型：这个技术有哪些功能，功能的接口是什么。

状态模型：系统在运行时有哪些状态，状态之间的变化原因是什么。

数据模型：这个技术是怎么组织数据的，是怎么处理数据的。

线程模型：这个技术有哪些线程，分别是做什么的。

构建这个技术骨架，我大部分都是靠自己的业余时间来完成的。在我日常工作中，如果统计下来算的话，应该有 30% 到 50% 的时间在学习。其中很大一部分都是在学习一些细碎的小东西，比如某个 API 或者某个 library 怎么用之类的。这些东西，都是技术骨架上的点缀。有了自己的技术骨架，接下来就可以等着问题上门了。问题送上门一个，收割一个，so happy。

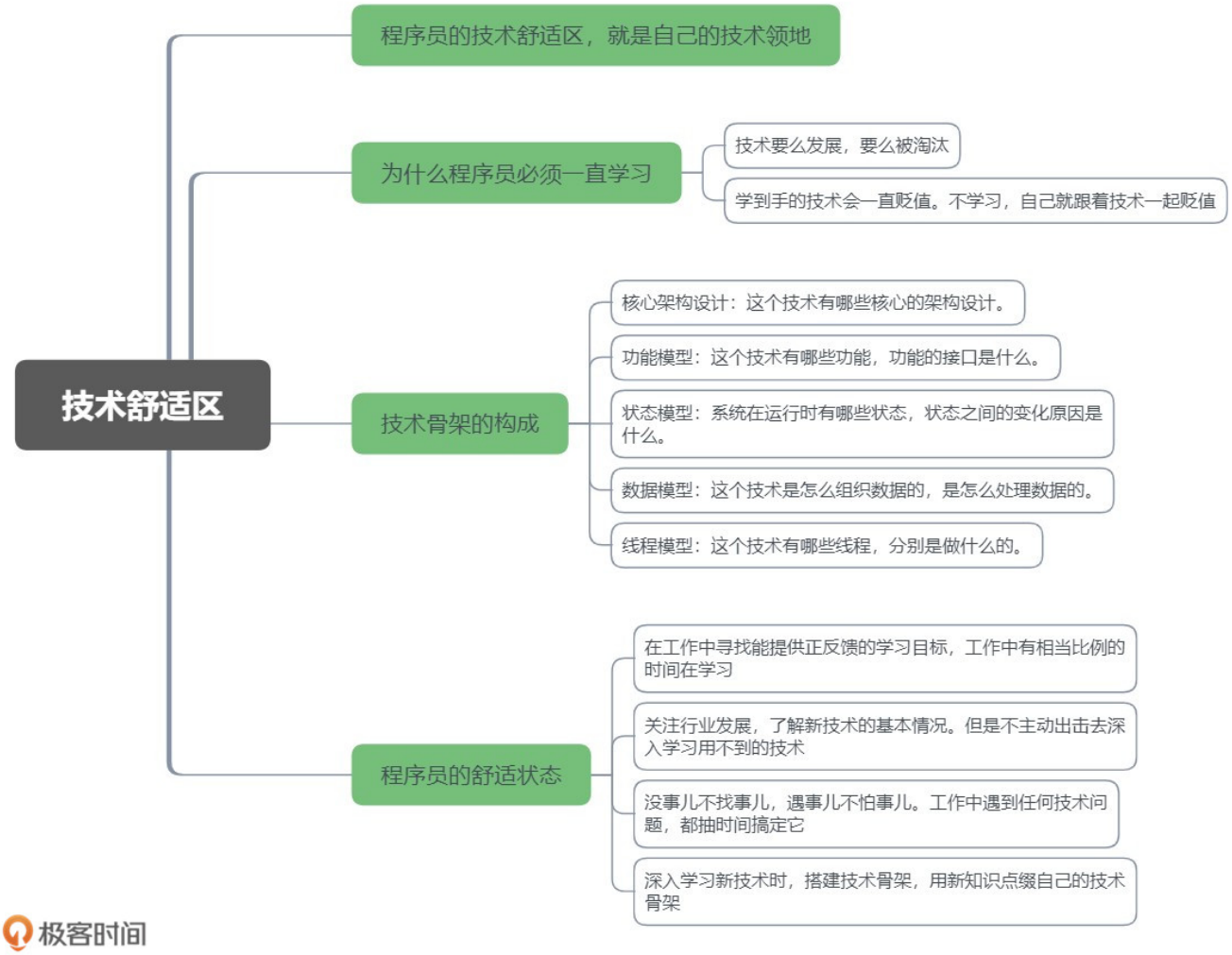
技术骨架的重要之处还在于，它一方面可以让我们学习相关的新东西时能抓住精要、触类旁通、学得更快更准。另一方面，技术骨架可以让我们学的知识成系统成生态，这样我们才能记得更牢固，用得更顺手。

总结

程序员的舒适区，其实就是自己的技术领地。技术领地越大，舒适区越大。当你精通 Java 了，Java 就是你的舒适区。当你听到一个 Java 的问题，心里就很舒适，因为你知道这对你来说根本不是问题。

技术只有被淘汰和正在进步两种状态。即使是网络，操作系统这些基础，也在不断地进步。随着 IPv6 的普及，原来 IPv4 的知识就被淘汰了。随着计算机硬件的发展，操作系统里的 CGroup 也应运而生，可以让我们更好地管理资源的分配。即使你原来精通网络，精通操作系统，如果几年不学习，回过头来也会发现自己的知识已经过时了。

所以说，程序员就是要在自己的旧领地被慢慢淘汰的同时，积极地去开拓自己的新领地，让自己的舒适区不断扩大。



极客时间

思考题

这篇文章，可以说是我这个懒人关于“如何打造技术舒适区”的一套方法。你是如何学习新技术的呢？哪些技术是你领地呢？哪些技术让你觉得自己可以舒舒服服地解决相关问题呢？

欢迎你在评论区写下自己的思考，也欢迎把这篇文章分享给你的朋友或者同事，一起交流一下。

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 21 | 答疑篇：想升职，我该准备些什么？

下一篇 23 | 技术观：做程序员，技术观为何如此重要？

精选留言 (7)

写留言

Geek_3b1096
2020-07-06

正在学习老师java课目标产生技术骨架

展开

作者回复:
(-: 那门课的骨架内容偏少，等我抽出时间来，会增加一些进阶偏骨架的内容~

3

牛牛
2020-07-06

是这样的、哈哈、努力让自己一直保持在舒适区~

不过、最初（一个新知识点的学习肯定是痛苦的）、熬过了这痛苦就会发现惊喜、持续不断的形成正反馈、就越来越有学习的动力了~

...

展开

作者回复: 数据结构， 算法，操作系统，网络，计算机系统结构这些都是内功。内功虽然没有立竿见影的效果，但是会对所有的能力都有加成。而且对你还没学的东西都有加成，因为你会发现学习这些东西会很快。

举个简单的例子，HBase里用一个叫做Bloom Filter的数据结构去做过滤，如果你知道，就会会心一笑，也知道这种过滤的效果和优缺点。如果你不知道，就懵了。这种例子不胜枚举

学好基础，以后学习就一帆风顺。基础不行，学习就是步步维艰。

不过你说的那些小惊喜呀，以后会越来越少，因为你都已经懂了哈哈哈哈哈

1 3

pyhhou
2020-07-07

老师能否详细讲一下如何构建自己的技术骨架？特别是对于刚入行 1，2 年的程序员，拿

我自己举例，首先因为经验并不是那么足，接触的技术并不多，也就没有什么横向的技术对比能力。另外老师说的技术骨架应该比较适用于类似 Spring 这样的框架类技术？对于一些比较基础或是比较细碎的技术，像是操作系统，网络，另外就是各种编程语言特性，对于这些知识的学习该如何构建出框架呢？问题有点多，谢谢老师

展开

作者回复:

我觉得这是一个很好的问题，我说的不是很详细。确实技术骨架不是一开始就能搞起来的，尤其是在自己的知识存量还没有形成规模效应，可以让我们触类旁通的时候。

我从开荒的状态说起吧，以Spring为例子。

首先肯定是要学好Java，明白对象，类，方法，多态，继承等核心概念。对于Spring来说，慢慢还要知道注解，反射，代理这些Spring会重点用到的知识。

然后就是学Spring，和学一般的技术一样，找个趁手的书或者资料，一以贯之的啃完。可以是我们的课程，也可以是官方文档。这个过程，开始更多的是比着葫芦画瓢，先看效果。

接着，就可以发挥自己的想象力来摸索了。这时候的最好的方式就是多写程序，想到什么就拿出来写写，练练，验证自己的想法。这一步，就是在慢慢构建自己的技术骨架了。所谓骨架，就是藏在表面之下的东西，看不到，但是如果没有它们支撑的话，表面的东西就不牢固。偏骨架的内容都在文章中都有介绍，这里就不重复了。就Spring来说，你要能知道这些bean是如何被描述的，如何被创建的，如何被组装的。做到这些，当出了某种异常的时候，你就很容易是哪个步骤的问题，可以马上去修正。

如果能做到这些，让你自己写一个IoC的框架，也非难事。当然，并不是说一定要做的像Spring这么全面，可能只是一个微小的内核，但是相信我，如果你自己做一遍，思考一遍，你会对Spring的认识更近一层，当你对比你自己的设计和实现与Spring的设计和实现时，会记的更牢固，印象更深刻。这个步骤，可以帮助你完整的构建关于这个技术的骨架。

当然，也不是说每个东西都要自己照着写一遍。如果自己实际做一遍，会对这个技术的核心和关键点认识的更深刻，当自己在做的过程中，遇到不好解决的问题时，可以参考现有的设计和实现。即使不去自己做一遍，而是在自己脑子里思考一下，如果自己做会如何设计，如何架构，想不通的地方去仔细看看现有的设计，也是能收获很多的。

1

Sdylan

2020-07-06

上周买了课没有看，这周开始纳入阅读计划。程序员的舒适区和我最近2年的想法简直一模一样。后面我也尝试搭建骨架，但我的还是那种老套路，基础模块（语言、OS、算法等）、进阶（JVM、多线程）、高阶（微服务、KV数据库），很粗。可以加入具体的骨架模型：核心模块、核心技术、数据模型、多线程模型。

...

展开

作者回复: 限于篇幅和内容流畅性，很多内容没展开说。

其实每种技术都有侧重性，有些技术可能也需要这几种模型之外的模型。比如偏向数据存储和处理的，重点考虑它的架构，然后是数据模型和线程模型。把这几个搞清楚了，就知道这个玩意是怎么转的，出了问题可能是哪里的问题，也能知道它是否适合一个场景。

其实每个人都有自己的习惯，但是这都没关系哈，重要的是要有个技术骨架，这样才能把知识整合在一起，才能系统的学会新领域的知识。否则就容易什么东西都只会个表面，容易浮躁。

2 1

落曦

2020-07-06

应届大三，感觉只会老师讲的考古级的技术栈，好丢人，啊啊啊，好害怕，应该开始学哪些才能脱离这考古级的技术😥

展开

作者回复: 多关心业界新闻，多关心行业新技术，试着玩一玩。极客时间上的极客新闻和infoq就是很不错的信息源。

1

小辉辉

2020-07-17

一直以来只有学习才能带来安全感，一天不学就感觉落后了很多。但也走过不少弯路，没有系统的学习，现在慢慢的在形成体系了

展开

作者回复: 不成体系很容易学了忘，忘了接着学。成体系可以把知识穿起来，记得牢固，用的放心。

追风筝的人

2020-07-16

拥抱变化，这个时代唯一不变的就是变化

展开

作者回复: ✅，不只是程序员呀，现在能混到退休的职业越来越少了